《Docker 入门白皮书》

中关村在线

www.zol.com.cn

2015 年 08 月

目录

写在前面：放在两年前，你不认识 Docker 情有可原。但如果现在你还这么说，不好意思，我只能说你 OUT 了。你最好马上 get 起来，因为有可能你们公司很快就会引入 Docker。今天就和大家讨论讨论这个备受好评的应用，让我们来揭开他的真面目！

【一】从 dotCloud 到 Docker——低调奢华有内涵

1、追根溯源：dotCloud

时间倒回到两年前，有一个名不见经传的小公司，他的名字叫做：dotCloud。dotCloud 公司主要提供的是基于 PaaS（Platform as a Service，平台及服务）平台为开发者或开发商提供技术服务，并提供的开发工具和技术框架。

初创企业总是艰难的，dotCloud 也是一样。在 IBM，亚马逊，谷歌等大公司的挤压下，dotCloud 举步维艰。即使 2011 年拿到了 1000 万美元的融资，可和上述大公司比起来，也不过是杯水车薪。

随着开源的洪流袭来，在 2013 年 dotCloud 的创始人，28 岁的 Solomon Hykes 做了一个艰难的决定：将 dotCloud 的核心引擎开源！然而一旦这个基于 LXC（Linux Container）技术的核心管理引擎开源，dotCloud 公司就相当于走上了一条"不归路"。

可正是这个孤注一掷的举动，却带来了全球技术人员的热潮，众程序员惊呼：太方便了，太方便了。也正是这个决定，让所有的 IT 巨头也为之一颤。一个新的公司也随之出世，它就是：Docker。

2、Docker 出世：从 Docker0.1 到 Docker1.0

一个春秋，跨越了 Docker 的成名路。

在互联网时代，一夜成名早已不是什么新闻。Docker 这个技术公司，向我们证明了，成为一个"国际巨星"，只需要一个月。2013 年 2 月决定开源，到 2013

年 3 月 20 日发布 Docker0.1，只用了一个月的时间。

今后几乎每个一个月，Docker 都会发布一个版本。而 Docker0.1 的发布像是一个宣言，昭示着一个 Docker 正在以一个新兴容器领导者的姿态迈进。

正如我们所知，从 Docker0.1 到 Docker1.0，15 个月的时间，Docker 迅速成长。在 2014 年 6 月 9 日，Docker 团队宣布发布 Docker 1.0 版。

Docker1.0 下载地址：
http：//blog.Docker.com/2014/06/its-here-Docker-1-0/ 。

2014 年 6 月 9 日，1.0 版本发布官方声明

1.0 版本标志着 Docker 公司认为 Docker 平台已经足够成熟，并可以被应用到产品中（还提供了一些需要付费的支持选项）。

在这 15 个月中，Docker 共收到了超过 460 位贡献者的 8741 条改进建议，

Docker 的用心经营下社区十分活跃。可以说，Docker 的成功起于开源，发于社区。

一年的时间，使一个围绕着 Docker 的小型初创企业生态体系逐渐形成。Docker 先后赢得了 Google、微软、Amazon、VMware 等巨头的青睐，巨头们纷纷示意将保证自己平台与 Docker 容器技术的兼容性。微软还宣布来要推出面向 Windows 的 Docker 客户端。

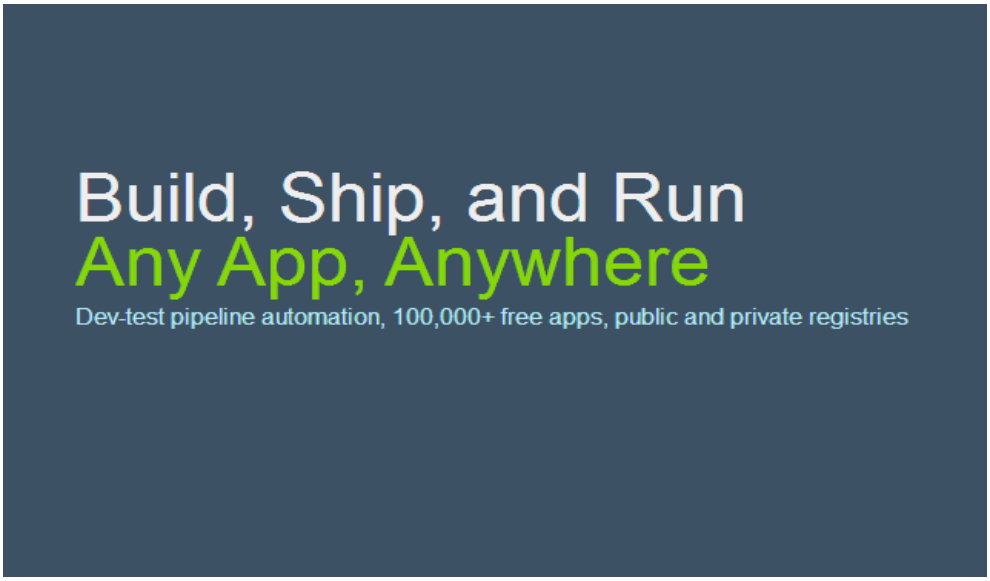
到了 2014 年 9 月，Docker 完成 4000 万美元的 C 轮融资，彼时市值与约为 4 亿美元。可以说 Docker 一路风生水起，迅速赢得了 IT 圈的信赖。并且在 8 月 12Docker 发布了 Docker1.8 正式版（下载地址见末尾）。

但 Docker 是如何做到这些的呢？Docker 的成功之路能否被复制呢？

【二】如何定义 Docker?

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上。

Docker 是一个重新定义了程序开发测试、交付和部署过程的开放平台，Docker 则可以称为构建一次，到处运行，这就是 Docker 提出的“Build once，Run anywhere”



Docker：Build once，Run anywhere

为了更好的认识 Docker ,我们先来了解几个必备词汇 :镜像 ,容器和仓库。

1、镜像：文件的层次结构，以及包含如何运行容器的元数据， Dockerfile 中的每条命令都会在文件系统中创建一个新的层次结构 ,文件系统在这些层次上构建起来，镜像就构建于这些联合的文件系统之上。

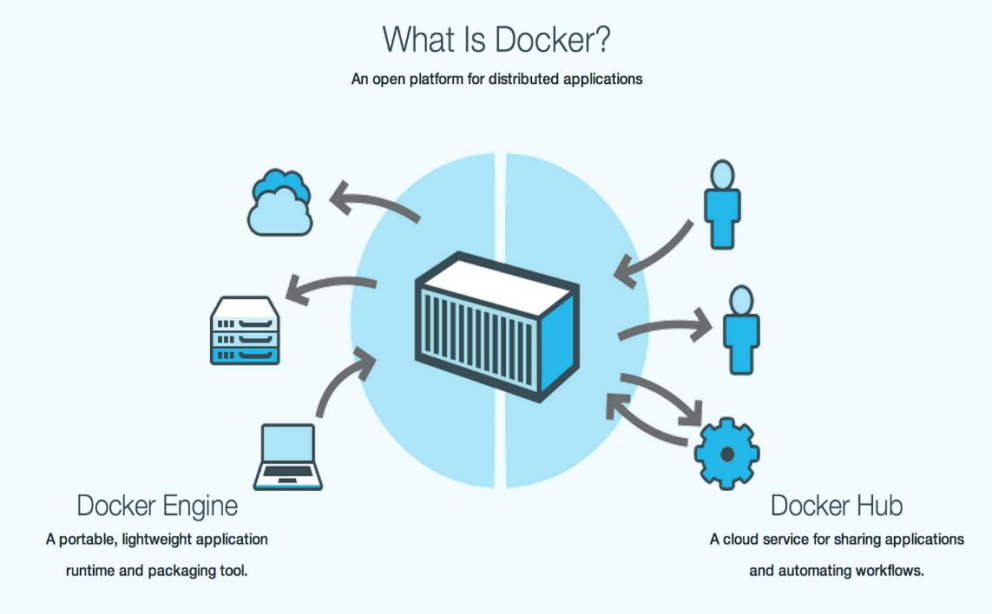
Docker 镜像就是一个只读的模板，镜像可以用来创建 Docker 容器。Docker 提供了一个很简单的机制来创建镜像或者更新现有的镜像 ,用户甚至可以直接从其他人那里下载一个已经做好的镜像来直接使用。

2、容器：容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。可以把容器看做是一个简易版的 Linux 环境，Docker 利用容器来运行应用。

3、仓库：仓库是集中存放镜像文件的场所，仓库注册服务器（Registry）上往往存放着多个仓库 ,每个仓库中又包含了多个镜像 ,每个镜像有不同的标签（tag）。目前，最大的公开仓库是 Docker Hub，存放了数量庞大的镜像供用户下载。

Docker 仓库用来保存我们的 images ,当我们创建了自己的 image 之后我们就可以使用 push 命令将它上传到公有或者私有仓库 ,这样下次要在另外一台机器上使用这个 image 时候，只需要从仓库上 pull 下来就可以了。

Docker 的运行离不开这几位的支持，Docker 的成功也是拜几位所赐。也有人会误以为，Docker 就是容器。*但 Docker 只会傲娇地说：我不是容器，我是管理容器的引擎。*



什么是 Docker

Docker 中文手册上解释说：Docker 是一个开源的引擎，可以轻松的为任何应用创建一个轻量级的、可移植的、自给自足的容器。开发者在笔记本上编译测试通过的容器可以批量地在生产环境中部署，包括 VMs（虚拟机）、bare metal、OpenStack 集群和其他的基础应用平台。

从这里我们可以看出，Docker 并非是容器，而是管理容器的引擎。Docker 为应用打包、部署的平台，而非单纯的虚拟化技术。

【三】Docker 与虚拟化争锋

	Docker容器	虚拟机（VM）
操作系统	与宿主机共享OS	宿主机OS上运行虚拟机OS
存储大小	镜像小，便于存储与传输	镜像庞大（vmdk、vdi等）
运行性能	几乎无额外性能损失	操作系统额外的CPU、内存消耗
移植性	轻便、灵活，适应于Linux	笨重，与虚拟化技术耦合度高
硬件亲和性	面向软件开发者	面向硬件运维者
部署速度	快速，秒级	较慢，10s以上

容器技术与传统虚拟机性能对比

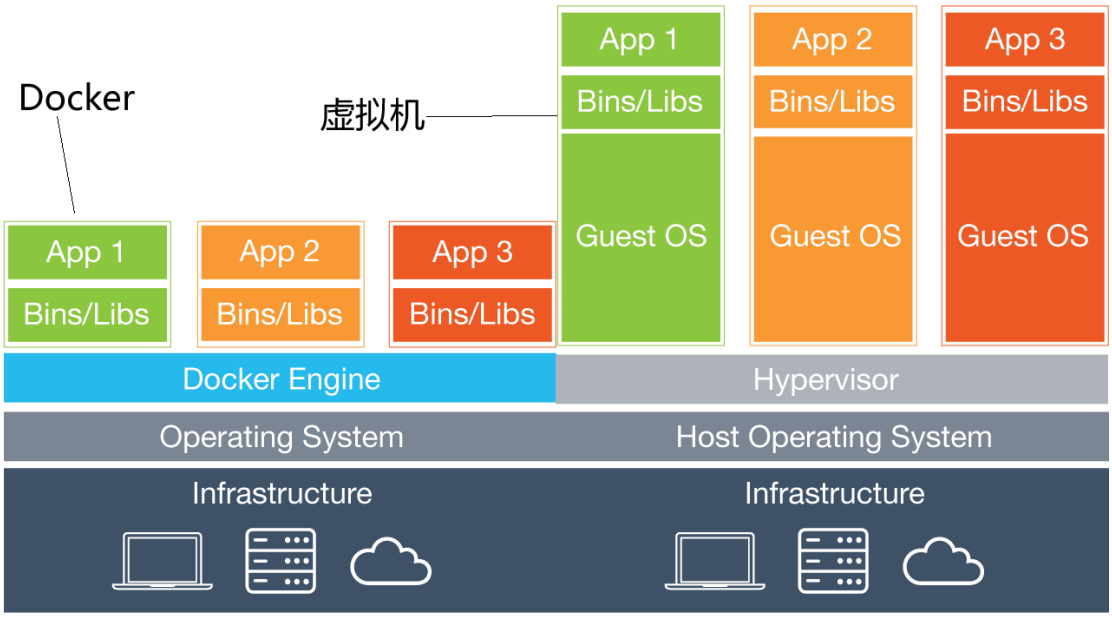
谈到虚拟化，很多人又发问了。Docker 和虚拟化有什么区别？Docker（或

者说是容器）的出现是否会取代传统的虚拟化技术。

说起虚拟化，大家首先想到的必然是 VM 一类的虚机。这类虚拟机完美的运行了另一套系统，能够使应用程序，操作系统和硬件三者之间的逻辑不变。

但在惜时如金的现在，这类虚机也面临着一定的问题，比如：启动时间太长，你有没有过在启动虚拟机后，点开其他页面继续操作，过了一分钟才回来的经历？还有虚拟镜像体积太大（一般都是几十 GB）等问题。相比之下，Docker 的镜像一般只有二三百兆。并且启动速度超快， Docker 的启动时间为毫秒级。

还有一个最大的问题是价格问题，据 StackEngine 调查分析，有 43.8%的企业使用 Docker 的原因是 VMvire 太贵。



Docker 与虚拟机建构对比

但是，传统的虚拟技术还不会被取代。Docker 或者说容器技术和虚拟机并非简单的取舍关系。

目前，很多企业仍在使用虚拟机技术，原因很简单，他们需要一个高效，安全且高可用的构架。然而，刚刚面世两年的 Docker 还没有经历沙场考验，CaaS（Container as a Service，容器即服务）概念也是近两年才刚刚出现。无论是应用管理还是运行维护方面，Docker 都还处于发展与完善阶段。

【四】Docker：我为什么与众不同

Solomon Hykes：成功的要素之一是在正确的时间做了正确的事，我们一直坚信这个理念。Docker 就好比传统的货运集装箱。但是创新可不仅仅是在这个盒子里 ,而且还包括如何自动管理呈现上万个这样的箱子。这才是问题的关键。

站在未来的角度，Docker 解决了三大现存问题。

Docker 让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，便可以实现虚拟化。

*俗话说：天下武学为快不破；在更新迭代如此之快的 IT 更是如此。*所有成功的 IT 公司都必须走在时代的前列，他们的产品应该来自未来。他们有必要要站在未来的角度解决现存的问题。

Docker 之父 Solomon Hykes：Docker 就好比传统的货运集装箱

Solomon Hykes 曾经说过，自己在开发 dotCloud 的 PaaS 云时，就发现一个让人头痛的问题 :应用开发工程师和系统工程师两者之间无法轻松协作发布产品。Docker 解决了难题。让开发者能专心写好程序；让系统工程师专注在应用的水平扩展、稳定发布的解决方案上。

1、简化程序：Docker 让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，便可以实现虚拟化。

Docker 改变了虚拟化的方式 ,使开发者可以直接将自己的成果放入 Docker 中进行管理。方便快捷已经是 Docker 的最大优势，过去需要用数天乃至数周的任务，在 Docker 容器的处理下，只需要数秒就能完成。

2、避免选择恐惧症：如果你有选择恐惧症，还是资深患者。Docker 帮你打包你的纠结！比如 Docker 镜像；Docker 镜像中包含了运行环境和配置，所以 Docker 可以简化部署多种应用实例工作。比如 Web 应用、后台应用、数据库应用、大数据应用比如 Hadoop 集群、消息队列等等都可以打包成一个镜像部署。

3、节省开支：一方面，云计算时代到来，使开发者不必为了追求效果而配置高额的硬件，Docker 改变了高性能必然高价格的思维定势。Docker 与云的结合，让云空间得到更充分的利用。不仅解决了硬件管理的问题，也改变了虚拟化的方式。

另一方面，Docker 能够是自愿额达到充分利用。举个简单地例子：凌晨三点的时候只有很少的人会去访问你的网站 ,同时你需要比较多的资源执行夜间的批处理任务，通过 Docker 可以很简单的便实现资源的交换。

Docker 的这些优势，让各大 IT 巨头纷纷对 Docker 看好。

【五】统一标准，建立更有活力的生态系统

开放容器技术项目（Open Container Project）

在 2015 年的 DockerCon 上推出了开放容器技术项目（Open Container Project）。OCP 是一个非营利性组织，其受特许建立通用的容器软件技术标准。

这个项目汇集了微软、谷歌、惠普、IBM、英特尔、红帽（Red Hat）、VMware 以及高盛等众多实力企业，OCP 的推出，也使宿敌 Docker 和 CoreOS 走向了联合。让微软与自己的竞争对手 Linux 合作，足以见得 Docker 的魅力。

谷歌云计算平台产品经理克雷格·麦克拉克伊(Craig Mcluckie)说：创建通用容器格式非常重要，单一标准可以促进更有活力的生态系统。

【六】企业对 Docker 是否认可？安全是关键！

随着容器技术逐渐得到 IT 界的认可，CaaS（Container as a Service，容器即服务）也逐渐形成。而 Docker 作为 CaaS 技术的标杆是否已经得到企业的认可？是否投入生产呢？

2015 年，VMblog.com 和 CloudCow.com 共同组织了一次问卷调查。报告显示，Docker 的早期用户中，63%的用于 QA/Test，53%的用于开发，并且 31%的用户计划在生产环境中使用 Docker，阻碍企业使用 Docker 的最大因素在于其安全性以及缺少生产环境下的运维工具（两个原因各占 49%左右）。

对 Docker 应用最广泛的三个领域分别是：Test/QA 应用；Web 应用；大数据，企业应用。

调查显示，目前企业对 Docker 的接受程度在不断提高。但 Docker 的安全性似乎仍旧是企业顾虑的主要原因，那么 Docker 的安全性究竟如何？

Gartner：Docker 还是一项年轻的技术，它的安全性仍不够成熟

2015 年 1 月，Gartner 分析师 Joerg Fritsch 发布一份报告，报告显示：虽然 Docker 这款容器化工具已经颇具名声，但 Docker 的安全性仍不够成熟。

Joerg Fritsch 指出："Docker 与容器技术目前还无法通过虚拟机管理程序弥合自身最为严重的两大短板：安全性保障与管理功能，外加在常见控制机制的机密性、完整性与可用性方面提供支持。"

总体来讲，Docker 的安全性能还不错，只是这还是一项年轻的技术、因此目前尚未积累起能够满足实际生产需求的完整工具生态系统。

其实如果要谈论 Docker 的安全性，我们就要谈论三点：命名空间（Namespace）；Docker 程序本身的抗攻击性和加固内核安全性来影响容器的安全性。

1、命名空间（Namespace）：Docker 有五个命名空间：进程、网络、挂载、宿主和共享内存，为了隔离有问题的应用，Docker 运用 Namespace 将进程隔离，为进程或进程组创建已隔离的运行空间，为进程提供不同的命名空间视图。这样，每一个隔离出来的进程组，对外就表现为一个 container(容器)。需要注意的是，Docker 让用户误以为自己占据了全部资源，但这并不是"虚拟机"。

内核 namesapce 从内核 2.6.15 之后被引入，距今已经 5 年了，在很多大型生产系统中被验证。他们的设计和灵感提出的时间更早，openvz 项目利用 namespace 重新封装他们的内核，并合并到主流内核中。openvz 最早的版本是 2005 年的，所以他们的设计和实现都很成熟。

2、Docker 程序本身的抗攻击性：Docker 允许你在主机和容器之间共享文件夹，这就容易让容器突破资源限制，那么容器就可以对主机做任何更改了。但实际上，几乎所有虚拟机系统都有在物理主机和虚拟机之间共享资源的限制，所以这一层的安全性，需要你自己把控。

3、加固内核安全性：默认情况下，Docker 启动的容器只使用一部分内核 capabilities，就算攻击者在容器中取得了 root 权限，他能做的破坏也少了，也不能获得主机的更高权限。

由此我们可以说：Docker 还是比较安全的，但是你要注意使用在容器中使用非 root 权限允许进程。

目前来说，Docker 的主要应用场景为：

面向开发人员 :快速开发、交付应用程序。开发环境的机器通常内存比较小，之前使用虚拟的时候，经常需要为开发环境的机器加内存，而现在 Docker 可以轻易的让几十个服务在 Docker 中跑起来。

面向运维人员：降低运维成本。正如通过虚拟机来整合多个应用，Docker 隔离应用的能力使得 Docker 可以整合多个服务器以降低成本。Docker 通过镜像机制，将你的代码和运行环境直接打包成镜像，扔到容器启动即可。

面向企业： Docker 本身就发家于 PaaS，在 Docker 面向企业，是可以提供 Paas 层的实现；比如，扩展现有的 OpenShift 或 Cloud Foundry 平台来搭建自己的 PaaS 环境。

Docker “集装箱”

【七】评说 Docker

Chris Swan（银行业的技术专家，曾经有十几年的时间在从事金融服务业）：Docker 公司已经建立了清晰的道路，即发展核心能力（libcontainer）、跨业务管理（libswarm）和容器间消息（libchan）。

Docker 公司表达了利用自身生态系统的意愿。随着时间的推移，虚拟机和容器（Docker 中的“运行”部分）之间的区别很可能变得不再那么重要，这将使注意力转到“构建（build）”和“交付（ship）”方面。

马全一（Docker 中文社区创始人之一）：Docker 一定会成为云计算和大数据领域的重要成员之一。

Docker 的出现使得以 Docker 容器为单位的云平台和 Docker 容器为载体的交易平台成为可能。任何后端的服务程序，都可以封装在 Docker 容器中进行销售、分发和部署，后端开发者能像 Mobile App 开发者那样去做自己的产品来获利。随着 Golang 的发展，一定会有以 Golang 为开发语言、Docker 为运行载体的新大数据平台，成为 Hadoop 平台的竞争者。

Solomon Hykes（Docker 之父）：“建立一个为所有生产软件共有的系统，使用一种被广泛接受的方式，让它可以很好的运行和扩展，让它可以被所有人依赖，然后将它视为理所当然的存在，并使用它创造自己的奇迹，这是个挑战。

Docker，一个迅速走红，并在技术领域赢得一片好评的一款应用。尽管 Docker 目前还不够完善，但已经有很多厂商已经开始使用。同时一批围绕 Docker 建立起来的初创企业已经形成，学习 Docker 风气正盛。可以预见在不远的未来，Docker 和 CaaS 会获得更多的人肯定，越来越多的企业愿意使用 Docker 这个应用。

据 Docker 内部人员表示：在 Docker 1.0 正式发布之前，就已经有 3 家主要银行将其投入到生产应用中

附录一 、Docker 基本命令

Docker ps 用来查看正在运行中的容器。

Docker ps 命令的常用参数（及组合）如下。

-a： 查看所有容器，包括已经停止运行的。

-l： 查看刚刚启动的容器。

-q： 只显示容器 ID

-l -q： 则可以返回刚启动的容器 ID。

Docker stop 用来停止运行中的容器，同时你还可以用 Docker start 来重新启动一个已经停止的容器。

Docker restart 可以重启一个运行中的容器。这就相当于对一个容器先进行 stop 再 start。

```
Options:

  --add-registry=[]                 Registry to query before a public one
  --api-cors-header=                Set CORS headers in the remote API
  -b, --bridge=                     Attach containers to a network bridge
  --bip=                            Specify network bridge IP
  --block-registry=[]               Don't contact given registry
  --confirm-def-push=true           Confirm a push to default registry
  -D, --debug=false                 Enable debug mode
  -d, --daemon=false                Enable daemon mode
  --default-gateway=                Container default gateway IPv4 address
  --default-gateway-v6=             Container default gateway IPv6 address
  --default-ulimit=[]               Set default ulimits for containers
  --dns=[]                          DNS server to use
  --dns-search=[]                   DNS search domains to use
  -e, --exec-driver=native          Exec driver to use
  --exec-opt=[]                     Set exec driver options
  --exec-root=/var/run/docker       Root of the Docker execdriver
  --fixed-cidr=                     IPv4 subnet for fixed IPs
  --fixed-cidr-v6=                  IPv6 subnet for fixed IPs
  -G, --group=docker                Group for the unix socket
  -g, --graph=/var/lib/docker       Root of the Docker runtime
  -H, --host=[]                     Daemon socket(s) to connect to
  -h, --help=false                  Print usage
  --icc=true                        Enable inter-container communication
  --insecure-registry=[]            Enable insecure registry communication
  --ip=0.0.0.0                      Default IP when binding container ports
  --ip-forward=true                 Enable net.ipv4.ip_forward
  --ip-masq=true                    Enable IP masquerading
  --iptables=true                   Enable addition of iptables rules
  --ipv6=false                      Enable IPv6 networking
  -l, --log-level=info              Set the logging level
  --label=[]                        Set key=value labels to the daemon
  --log-driver=json-file            Default driver for container logs
  --log-opt=map[]                   Set log driver options
  --mtu=0                           Set the containers network MTU
  -p, --pidfile=/var/run/docker.pid Path to use for daemon PID file
  --registry-mirror=[]              Preferred Docker registry mirror
  -s, --storage-driver=             Storage driver to use
  --selinux-enabled=false           Enable selinux support
  --storage-opt=[]                  Set storage driver options
  --tls=false                       Use TLS; implied by --tlsverify
  --tlscacert=~/.docker/ca.pem      Trust certs signed only by this CA
  --tlscert=~/.docker/cert.pem      Path to TLS certificate file
  --tlskey=~/.docker/key.pem        Path to TLS key file
  --tlsverify=false                 Use TLS and verify the remote
  --userland-proxy=true             Use userland proxy for loopback traffic
  -v, --version=false               Print version information and quit
```

```
Commands:
    attach    Attach to a running container
    build     Build an image from a Dockerfile
    commit    Create a new image from a container's changes
    cp        Copy files/folders from a container's filesystem to the host path
    create    Create a new container
    diff      Inspect changes on a container's filesystem
    events    Get real time events from the server
    exec      Run a command in a running container
    export    Stream the contents of a container as a tar archive
    history   Show the history of an image
    images    List images
    import    Create a new filesystem image from the contents of a tarball
    info      Display system-wide information
    inspect   Return low-level information on a container or image
    kill      Kill a running container
    load      Load an image from a tar archive
    login     Register or log in to a Docker registry server
    logout    Log out from a Docker registry server
    logs      Fetch the logs of a container
    pause     Pause all processes within a container
    port      Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
    ps        List containers
    pull      Pull an image or a repository from a Docker registry server
    push      Push an image or a repository to a Docker registry server
    rename    Rename an existing container
    restart   Restart a running container
    rm        Remove one or more containers
    rmi       Remove one or more images
    run       Run a command in a new container
    save      Save an image to a tar archive
    search    Search for an image on the Docker Hub
    start     Start a stopped container
    stats     Display a stream of a containers' resource usage statistics
    stop      Stop a running container
    tag       Tag an image into a repository
    top       Lookup the running processes of a container
    unpause   Unpause a paused container
    version   Show the Docker version information
    wait      Block until a container stops, then print its exit code
```

Docker 服务对应的版本查看

sudo Docker version

Docker 命令帮助

#sudo Docker　//查看 Docker 的所有命令

#sudo Docker command --help　//查看单个 Docker 命令的帮助 ,如 Docker ru --help

附录二、Docker 最新版（1.8）下载地址：

Ubuntu/Debian： curl -sSL https：//get.Docker.com | sh

Linux 64bit binary：
https：//get.Docker.com/builds/Linux/x86_64/Docker-1.8.0

Darwin/OSX 64bit client binary：
https：//get.Docker.com/builds/Darwin/x86_64/Docker-1.8.0

Darwin/OSX 32bit client binary：
https：//get.Docker.com/builds/Darwin/i386/Docker-1.8.0

Linux 64bit tgz：
https：//get.Docker.com/builds/Linux/x86_64/Docker-1.8.0.tgz

Windows 64bit client binary：
https：//get.Docker.com/builds/Windows/x86_64/Docker-1.8.0.exe

Windows 32bit client binary：
https：//get.Docker.com/builds/Windows/i386/Docker-1.8.0.exe

相关网站链接：Docker 官网：https://www.Docker.com/

Mac 安装方式：http://docs.Docker.com/mac/started/

Linux 安装方式：http://docs.Docker.com/linux/started/

Windows 安装方式：http://docs.Docker.com/windows/started/